Intelligens rendszerfelügyelet

2. házi feladat – Konfigurációkezelés

Feladat kódja: 2-D

Sajben Zsolt Tamás
2011.04.10.

1. A feladat rövid ismertetése

A feladat egy olyan Bash szkript, ami a bemeneti CSV állományban (1. paraméter) megadott gépekről lekérdezi CIM-XML használatával, hogy milyen felcsatolt fájlrendszerekkel rendelkeznek, ezután pedig a kimenetként (2. paraméter) megadott elérési útvonalra HTML formátumú jelentést készít a kapott eredményekről.

2. A feladat pontosítása és célok meghatározása

A bemeneti **paraméterek** közül mindkettő (bemeneti CSV, kimeneti HTML fájl) megadása kötelező. Bármelyik hiánya esetén a szkript hibát ír ki, és leáll. A futás szükséges feltétele továbbá az is, hogy az 1. paraméterként megadott elérési útvonalú állomány a szkript indulásakor létezzen, a 2.-ként megadott viszont ne létezzen.

A bemeneti **CSV fájl** a lekérdezendő gépeket, és az azokhoz való csatlakozás módját a következő oszlopokban adja meg:
`machineName,port,protocol,user,password`

A kimeneti **HTML állomány** <h1> szintű címsorral tartalmazza a gépek neveit, alattuk pedig minden felcsatolt fájlrendszerről 1-1 táblázatot, a következő attribútumokkal és értékükkel (ebben a sorrendben):

- csatolási pont
- teljes méret
- százalékos foglaltság
- fájlrendszer típusa

A szkript a **standard kimenet**re csak a hibaüzeneteket írja ki, ha szükséges, mást nem.

3. Az elvégzett munka részletes leírása

Első feladatunk, hogy leellenőrizzük, hogy mindkét paraméter meg lett-e adva, és hogy az 1. fájl létezik, a 2. nem. Ezt a `test` utasítással (szögletes zárójel) könnyen megtehetjük:

```
# parameterek szamanak ellenorzese
if [ $# -ne 2 ]; then
    echo "Usage: getMounts.sh <machineData> <outputFile>"
    exit 1
fi

# letezzen a bemeneti fajl
if [ ! -e $1 ]; then
    echo "File given by the 1st parameter should exist!"
    exit 2
fi

# ne letezzen a kimeneti fajl
if [ -e $2 ]; then
    echo "File given by the 2nd parameter shouldn't exist!"
    exit 3
fi
```

Ezután elkezdjük összerakni a majdani kimenetet egy változóba. Itt még csak a HTML fájl elejét készítjük el:

```
# a HTML fajl elejet osszeallitjuk egy sztringben, a
  tovabbiakban is ehhez fogunk hozzafuzni
result="<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01
  Transitional//EN\"\n\t\"http://www.w3.org/TR/html4/
  loose.dtd\">\n"
result="${result}<html>\n"
result="${result}<head>\n"
result="${result}\t<title>Mounted Filesystems</title>\n"
result="${result}\t<meta http-equiv=\"Content-Type\"
  content=\"text/html; charset=utf-8\">\n"
result="${result}\t<style type=\"text/css\">\n"
result="${result}\t\tbody {font-family: Cambria,
  Times New Roman, sans-serif}\n"
result="${result}\t\th1 {font-family: Times New Roman,
  sans-serif}\n"
result="${result}\t\ttable {border:1px solid black;
  width:300px; margin-top: 5px;}\n"
result="${result}\t\tth, td {border:1px solid black;
  padding: 2px;}\n"
result="${result}\t\tth {text-align:right; width:170px;}\n"
result="${result}\t</style>\n"
result="${result}</head>\n"
result="${result}<body>\n"
```

Majd a bemeneti fájl adatai számára tömböket készítünk, és feltöltjük őket a megadott adatokkal, az 1. paraméterként megadott állományból (mindig a „tömbhossz+1”-edik indexekre szúrjuk be a beolvasott adatokat):

```
# a megadott gepek adatainak tarolasara letrehozunk tomboket
mach=()
port=()
prot=()
user=()
pass=()

# beolvassuk az adatokat a bemeneti CSV fajlbol
while IFS=, read mach_ port_ prot_ user_ pass_
do
    mach[${${#mach[@]}+1]]=$mach_
    port[${${#port[@]}+1]]=$port_
    prot[${${#prot[@]}+1]]=$prot_
    user[${${#user[@]}+1]]=$user_
    pass[${${#pass[@]}+1]]=$pass_
done < $1
```

Egy ciklusban végigvesszük az összes megadott gépet. A ciklusszámláló 2-től indul, mert egyrészt a feltöltés módja miatt a 0. elemek mindig üresek lesznek (az `array[$[${#array[@]}+1]]` az első beolvasáskor a hossz+1 = 1-es indexű elemnek ad értéket), másrészt pedig a CSV első sorából kiolvasott értékek csak az oszlopok neveit tárolják, így a tömbök 1. elemével sem fogunk foglalkozni:

```
# minden megadott gepen vegigmegyunk
for ((i=2; i<=${#mach[@]}; i++))
do

    # ide jönnek a következő utasítások

done;
```

Ezen belül a következőket tesszük. Először is összeállítjuk a lekérdezendő URL-t a tömbjeink megfelelő indexű elemeiből:

```
# osszeallitjuk a lekerdezendo URL-t
command="${prot[$i]}://${user[$i]}:${pass[$i]}@${mach[$i]}:$
  {port[$i]}/root/cimv2:CIM_FileSystem"
```

Majd végrehajtjuk a lekérdezést, és egyből szűrünk azokra a sorokra, amik a számunkra szükséges attribútumokat tartalmazzák. A végeredményt egy tömbbe mentjük el. Látható, hogy a `grep`-nek átadott mintában escape karaktert („\”-t) kell tennünk a kötőjelek, és a részmintákat elválasztó VAGY jelek („|”) elé:

```
# vegrehajtjuk a lekerdezest es egybol szurunk is a
  szukseges elemekre
found=( $(wbemcli -nl ei $command | grep
  '\-Root=\|\-FileSystemSize=\|\-PercentageSpaceUse=\|
  \-FileSystemType=') )
```

Ellenőrizzük, hogy hiba nélkül lefutott-e az előző parancs (0 volt-e a visszatérési érték). Ha nem, a következő gépre ugrunk, a majdani HTML-be ilyen esetben semmit sem kell beleírni:

```
# ha sikertelen volt, a kovetkezore ugrunk
if [ $? -ne 0 ]; then
    continue
fi
```

Ha nem volt hiba, akkor pedig továbbhaladunk. A kiírandó karakterlánchoz hozzáadjuk az aktuálisan lekérdezett címet „h1”-es címsorként:

```
# ha sikeres volt
# a kiirando sztringhez hozzaadjuk a gep cimet
result="${result}<h1>${mach[$i]}</h1>\n"
```

Tömböket készítünk a lekérdezett géphez csatolt fájlrendszerek adatainak csoportosított tárolására. (Egyből kiírhatnánk minden adatot a következő, beágyazott ciklusban, de úgy nem jó sorrendben lennének a táblázatok sorai, ezért kell ez a plusz trükk, hogy a következő ciklusban még csak eltároljuk őket, és majd az után olvassuk ki az adatokat, sorrendhelyesen.)

```
# az ehhez a gephez tartozo fajlrendszerek adatai,
  csoportositva:
    roots=()
    sizes=()
    spaces=()
    types=()
```

Most jön az előbb említett beágyazott ciklus, amelyben feldolgozzuk a lekérdezéskor megkapott tömb elemeit (kiolvassuk belőlük az adott attribútum nevét és értékét).

```
# a kapott tombbol kiszedjuk a nev-ertek parokat, kulon-
  kulon
for ((j=0; j<${#found[@]}; j++));
do

    # ide jönnek a következő utasítások

done;
```

Ezen belül a következőket kell tenni. Először is a ciklusszámlálót el kell osztani 4-gyel, és a végeredményből egész számot kell készíteni, hogy megtudjuk, hogy az adott gép hányadik fájlrendszerének adatával foglalkozunk (mindegyikhez 4 adatot tárolunk, ezért kell 4-gyel osztani). Ennek a megvalósítása a következő:

```
# kiszamoljuk, hogy  hanyadik fajlrendszerrol van szo
nr=$((j / 4))
nr=${nr/\.*}    # levagjuk a veget, hogy egesz szam legyen
  belole
```

Kiszámoljuk a kapott érték 2-szeresét, és az annál 1-gyel nagyobb számot, ugyanis ezek többször fognak még kelleni (ne kelljen mindig kiszámolni):

```
# kelleni fognak a kovetkezo belole szarmaztatott ertekek:
nr2=$((nr * 2))
nr21=$((nr2 + 1))
```

Most jön a beágyazott ciklus lényege, az adatok kiolvasása a lekérdezéskor előállított tömbökből. Ehhez `awk`-t használunk (az „=” jelek mentén darabolunk):

```
# kibanyasszuk az adatokat az adott tombelembol
name=`echo ${found[$j]} | awk -F= '{print $1}'`
value=`echo ${found[$j]} | awk -F= '{print $2}'`
# a nev elejerol elhagyjuk a kotojelet
name=${name:1}
```

A fenti adatokat egy `case` szerkezetben a megfelelő tömbökbe mentjük (a nevet és az értéket 2 egymás utáni elembe):

```
# berakjuk a nev-ertek parost a neki megfelelo tomb 2 egymas
  utani elemebe
case "$name" in
    Root)
        roots[$nr2]=$name
        roots[$nr21]=$value
        ;;
    FileSystemSize)
        sizes[$nr2]=$name
        sizes[$nr21]=$value
        ;;
    PercentageSpaceUse)
        spaces[$nr2]=$name
        spaces[$nr21]=$value
        ;;
    FileSystemType)
        types[$nr2]=$name
        types[$nr21]=$value
        ;;
    .*)
        echo "error"
        exit 4
        ;;
esac
```

Itt véget ér az eddigi beágyazott ciklus, és egy új kezdődik (ebben a feldolgozott adatokat immár sorrendhelyesen bele tudjuk írni a „`result`”-ba). A tömbökön 2 elemenként megyünk végig, hiszen így tároltuk el bennük az értékeket:

```
# vegigmegyunk a keszitett tombokon (most mar
  sorrendhelyesen ki tudjuk irni a parosokat)
for ((j=0; j<$((${#roots[@]} / 2)); j++));
do

    # ide jönnek a következő utasítások

done;
```

Ezen belül mindig először kirakunk egy `<table>` tag-et:

```
# rakunk egy <table> taget
result="${result}\t<table>\n"
```

Aztán az előző ciklushoz hasonlóan, kiszámolunk néhány, később szükséges származtatott értéket:

```
# ujabb szarmaztatott ertekek, a keszitett tombok
  bejarasahoz
j2=$((j * 2))
j21=$((j2 + 1))
```

Majd a `result`-hoz hozzáfűzzük a tábla sorait, sorrendhelyesen:

```
# beleirjuk a tablaba a sorokat, a nev-ertek parosokkal
result="${result}\t\t<tr>\n\t\t\t<th>${roots[$j2]}</th>\n
  \t\t\t<td>${roots[$j21]}</td>\n\t\t</tr>\n"
result="${result}\t\t<tr>\n\t\t\t<th>${sizes[$j2]}</th>\n
  \t\t\t<td>${sizes[$j21]}</td>\n\t\t</tr>\n"
result="${result}\t\t<tr>\n\t\t\t<th>${spaces[$j2]}</th>\n
  \t\t\t<td>${spaces[$j21]}</td>\n\t\t</tr>\n"
result="${result}\t\t<tr>\n\t\t\t<th>${types[$j2]}</th>\n
  \t\t\t<td>${types[$j21]}</td>\n\t\t</tr>\n"
```

Végül pedig lezárjuk a táblát:

```
# irunk egy </table>-t
result="${result}\t</table>\n"
```

Itt véget ér a beágyazott ciklus, és a külső ciklus is. A futás végén még kiegészítjük a kiírandó karakterláncot, hogy le legyenek zárva a HTML tag-ek:

```
# a HTML vege
result="${result}</body>\n"
result="${result}</html>"
```

Végül pedig kiírjuk az eddig összerakott karakterfüzért a szkriptnek megadott 2. paraméternek megfelelő elérési útvonalra:

```
#kiiras a fajlba
echo -e $result > $2
```

4. Értékelés, összefoglalás és tanulságok

A feladat hasznos tapasztalatokkal gazdagított, hiszen olyan programot írtam, ami alapvetően hordozható Linux-ok között (nem tudom pontosan, mennyi részlet van, amit át kéne írni, ha pl. egy Ubuntun akarnám futtatni, de gyanítom, hogy (szinte) semmit, hiszen azok a szintaxisú utasítások, amiket a megoldás készítése közben az Interneten találtam, nem voltak CentOS specifikusak).

Fontos tanulság volt számomra, hogy a különböző zárójelek, idézőjelek, és dollár jelek („$”) alkalmazásakor nagyon figyeljek oda, hiszen a legkisebb elírásokat tartott a legtovább kijavítani (pl. értékadáskor az a változó, aminek értéket adunk, az elé nem kell „$” – egy ilyen hibára nagyjából fél órám ment rá, mire rájöttem, hogy hol van az elírás).

5. Megjegyzések

- **Fontos!** A CSV fájl utolsó sorának a végén is kell, hogy legyen sortörés, különben a szkript az utolsó gépet ki fogja hagyni, nem fogja lekérdezni annak felcsatolt fájlrendszereit.
- Ha a paraméterek száma több, mint 2, azt nem vettem problémának. Ilyenkor nem ír ki hibát, és nem áll le a futás, egyszerűen csak nem használja az első két paraméteren kívül megadott értékeket.
- A kódban a kommentek szövege magyar, de a kiírások és a változónevek angol nyelvűek.

6. Fejlesztési- és tesztkörnyezet

A kiadott CentOS virtuális gép két példányát használtam, egy Windows Vista operációs rendszerrel rendelkező gépen, VMware Player segítségével.

Mindkét gépen a következőket végeztem el:

- A hálózati kapcsolatot „Bridged” módba állítottam, hogy az otthoni LAN-on lévő többi géppel is el tudjam érni őket.
- Elindítottam rajtuk az OpenPegasus-t (hogy mindkettőt le tudjam kérdezni):
  ```
  sudo /etc/init.d/tog-pegasus start
  ```

7. Tesztelés

7.1 Paraméterek

Teszteltem, hogy a paraméterekre vonatkozó feltételek megsértése esetén kilép-e hibával a program. (Ilyenkor kimeneti fájlnak sem szabad keletkeznie.)

7.1.1 Hiányzó paraméterek

A következők nem szabad, hogy lefussanak:

```
./getMounts.sh
./getMounts.sh abc
```

Ezt a hibát írják ki:

```
Usage: getMounts.sh <machineData> <outputFile>
```

7.1.2 Nem létező bemeneti fájl

Szintén hibával leáll (tegyük fel, hogy „abc” nevű fájl nem létezik):
`./getMounts.sh abc def`

Ezt a hibát írja ki:
`File given by the 1st parameter should exist!`

7.1.3 Létező kimeneti fájl

Szintén hibával leáll (tegyük fel, hogy „in.csv” és „out.html” nevű fájl is létezik):
`./getMounts.sh in.csv out.html`

Ezt a hibát írja ki:
`File given by the 2nd parameter shouldn't exist!`

7.2 Egy komplex teszteset

Teszteléskor mindig ugyanazt a komplex tesztesetet hajtottam végre, ami magában foglal minden, az otthoni feltételek mellett letesztelhető kapcsolódási esetet:

A kiadott **parancs** („in.csv” létezik, „out.html” nem):
`./getMounts.sh in.csv out.html`

Az „**in.csv**” tartalma (az utolsó sor végén is van sortörés!):

```
machineName,port,protocol,user,password
192.168.1.6,5988,http,pegasus,LaborImage
localhost,5989,https,pegasus,LaborImage
192.168.1.7,5988,http,pegasus,LaborImage
192.168.1.7,5989,https,pegasus,LaborImage
```

A **keletkezett „out.html” fájl** megfelelő:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
       "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
       <title>Mounted Filesystems</title>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
       <style type="text/css">
              body {font-family: Cambria, Times New Roman, sans-serif}
              h1 {font-family: Times New Roman, sans-serif}
              table {border:1px solid black; width:300px; margin-top: 5px;}
              th, td {border:1px solid black; padding: 2px;}
              th {text-align:right; width:170px;}
       </style>
</head>
<body>
<h1>192.168.1.6</h1>
       <table>
              <tr>
                     <th>Root</th>
                     <td>"/"</td>
              </tr>
              <tr>
                     <th>FileSystemSize</th>
                     <td>4030406656</td>
              </tr>
```

```
		<tr>
			<th>PercentageSpaceUse</th>
			<td>49</td>
		</tr>
		<tr>
			<th>FileSystemType</th>
			<td>"ext3"</td>
		</tr>
	</table>
	<table>
		<tr>
			<th>Root</th>
			<td>"/boot"</td>
		</tr>
		<tr>
			<th>FileSystemSize</th>
			<td>103512064</td>
		</tr>
		<tr>
			<th>PercentageSpaceUse</th>
			<td>36</td>
		</tr>
		<tr>
			<th>FileSystemType</th>
			<td>"ext3"</td>
		</tr>
	</table>
<h1>localhost</h1>
	<table>
		<tr>
			<th>Root</th>
			<td>"/"</td>
		</tr>
		<tr>
			<th>FileSystemSize</th>
			<td>4030406656</td>
		</tr>
		<tr>
			<th>PercentageSpaceUse</th>
			<td>49</td>
		</tr>
		<tr>
			<th>FileSystemType</th>
			<td>"ext3"</td>
		</tr>
	</table>
	<table>
		<tr>
			<th>Root</th>
			<td>"/boot"</td>
		</tr>
		<tr>
			<th>FileSystemSize</th>
			<td>103512064</td>
		</tr>
		<tr>
			<th>PercentageSpaceUse</th>
			<td>36</td>
		</tr>
		<tr>
			<th>FileSystemType</th>
			<td>"ext3"</td>
		</tr>
	</table>
```

```
<h1>192.168.1.7</h1>
	<table>
		<tr>
			<th>Root</th>
			<td>"/"</td>
		</tr>
		<tr>
			<th>FileSystemSize</th>
			<td>4030406656</td>
		</tr>
		<tr>
			<th>PercentageSpaceUse</th>
			<td>49</td>
		</tr>
		<tr>
			<th>FileSystemType</th>
			<td>"ext3"</td>
		</tr>
	</table>
	<table>
		<tr>
			<th>Root</th>
			<td>"/boot"</td>
		</tr>
		<tr>
			<th>FileSystemSize</th>
			<td>103512064</td>
		</tr>
		<tr>
			<th>PercentageSpaceUse</th>
			<td>36</td>
		</tr>
		<tr>
			<th>FileSystemType</th>
			<td>"ext3"</td>
		</tr>
	</table>
<h1>192.168.1.7</h1>
	<table>
		<tr>
			<th>Root</th>
			<td>"/"</td>
		</tr>
		<tr>
			<th>FileSystemSize</th>
			<td>4030406656</td>
		</tr>
		<tr>
			<th>PercentageSpaceUse</th>
			<td>49</td>
		</tr>
		<tr>
			<th>FileSystemType</th>
			<td>"ext3"</td>
		</tr>
	</table>
	<table>
		<tr>
			<th>Root</th>
			<td>"/boot"</td>
		</tr>
		<tr>
			<th>FileSystemSize</th>
			<td>103512064</td>
		</tr>
```

```
                <tr>
                        <th>PercentageSpaceUse</th>
                        <td>36</td>
                </tr>
                <tr>
                        <th>FileSystemType</th>
                        <td>"ext3"</td>
                </tr>
        </table>
</body>
</html>
```

7.3 Helytelen adatok a CSV-ben

Ha egy, a CSV-ben megadott géphez a szkript **nem tud csatlakozni** a protokoll, a gépnév, vagy a portszám helytelensége miatt, a következőt írja ki a `wbemcli` parancs, és így a szkript:

```
 *
 * wbemcli: Http Exception: couldnt't connect to server
 *
```

Ha tud csatlakozni, de a **portszám és a protokoll nem illik össze**, akkor az alábbi kettő közül valamelyiket:

```
 *
 * wbemcli: Http Exception: SSL connect error
 *
```

```
 *
 * wbemcli: Http Exception: server returned nothing (no
  headers, no data)
 *
```

Ha pedig az eddig felsoroltakkal nincs gond (a megadott gépen, a megadott porton keresztül, a megadott protokollal elérhető egy CIM szerver), de a **felhasználónév vagy jelszó nem megfelelő**, a következő hibát kapjuk:

```
 *
 * wbemcli: Http Exception: Invalid username/password.
 *
```

A felsorolt esetekben a kimeneti HTML állományba az adott gép, amelyiket nem sikerült lekérdezni, természetesen nem kerül bele.

Ezeket a következőképp **teszt**eltem le:
`./getMounts.sh in2.csv out2.html`

Az „**in2.csv**” tartalma (hibák kiemelve):

```
machineName,port,protocol,user,password
192.168.1.111,5988,http,pegasus,LaborImage
localhost,5988,https,pegasus,LaborImage
localhost,5989,http,pegasus,LaborImage
localhost,5988,http,asd,LaborImage
localhost,5988,http,pegasus,asd
localhost,5988,http,pegasus,LaborImage
```

Vagyis:

- 1. sor: gépnév helytelen
- 2. sor: port és protokoll nem illik össze
- 3. sor: szintén a port és a protokoll nem illik össze, de itt más a kiírt hiba
- 4. sor: a felhasználónév nem megfelelő
- 5. sor: a jelszó nem megfelelő
- 6. sor: helyes

Erre a kimenet a **konzol**on az elvártak szerint alakul (5 hiba, csak a CSV utolsó sora érvényes):

```
 *
 * wbemcli: Http Exception: couldnt't connect to server
 *
 *
 * wbemcli: Http Exception: SSL connect error
 *
 *
 * wbemcli: Http Exception: server returned nothing (no
  headers, no data)
 *
 *
 * wbemcli: Http Exception: Invalid username/password.
 *
*
 * wbemcli: Http Exception: Invalid username/password.
 *
```

Az „**out2.html**” is megfelelő, csak egy gép jelenik meg benne:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
       "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
       <title>Mounted Filesystems</title>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
       <style type="text/css">
              body {font-family: Cambria, Times New Roman, sans-serif}
              h1 {font-family: Times New Roman, sans-serif}
              table {border:1px solid black; width:300px; margin-top: 5px;}
              th, td {border:1px solid black; padding: 2px;}
              th {text-align:right; width:170px;}
       </style>
</head>
<body>
<h1>localhost</h1>
       <table>
              <tr>
                     <th>Root</th>
                     <td>"/"</td>
              </tr>
              <tr>
                     <th>FileSystemSize</th>
                     <td>4030406656</td>
              </tr>
              <tr>
                     <th>PercentageSpaceUse</th>
                     <td>49</td>
              </tr>
              <tr>
                     <th>FileSystemType</th>
                     <td>"ext3"</td>
              </tr>
       </table>
       <table>
              <tr>
                     <th>Root</th>
                     <td>"/boot"</td>
              </tr>
              <tr>
                     <th>FileSystemSize</th>
                     <td>103512064</td>
              </tr>
              <tr>
                     <th>PercentageSpaceUse</th>
                     <td>36</td>
              </tr>
              <tr>
                     <th>FileSystemType</th>
                     <td>"ext3"</td>
              </tr>
       </table>
</body>
</html>
```

8. Hivatkozások

A következő oldalakról merítettem ötleteket a megoldáshoz:

- http://stackoverflow.com/questions/2576622/bash-assign-grep-regex-results-to-array
- http://www.cyberciti.biz/faq/finding-bash-shell-array-length-elements/#comments
- http://www.justlinux.com/forum/showthread.php?t=140388
- http://tldp.org/LDP/abs/html/ops.html
- http://www.linuxquestions.org/questions/programming-9/split-function-bash-script-7880/
- http://www.thegeekstuff.com/2010/07/bash-string-manipulation/
- http://www.linuxquestions.org/questions/linux-general-1/line-break-on-bash-script-617324/
- http://www.unix.com/shell-programming-scripting/31608-read-csv-file-assign-values-variable.html
- http://www.linuxquestions.org/questions/linux-newbie-8/bash-variable-array-trying-to-add-another-value-into-the-array-692226/
- http://www.thegeekstuff.com/2010/07/bash-case-statement/
- http://www.linuxquestions.org/questions/programming-9/convert-float-to-integer-in-bash-468503/
- http://www.linuxquestions.org/questions/programming-9/converting-string-to-integer-in-bash-608444/
- http://www.unix.com/shell-programming-scripting/107334-how-use-catch-try-final-bash-script.html